

Computing Real Numbers using DNA Self-Assembly

Shalin Shah, Parth Dave and Manish K Gupta
Email :- {shah_shalin, dave_parth, mankg}@daiict.ac.in

Laboratory of Natural Information Processing,
Dhirubhai Ambani Institute of Information and Communication Technology,
Gandhinagar 382007, Gujarat, India

Abstract. DNA Self-Assembly has emerged as an interdisciplinary field with many intriguing applications such as DNA bio-sensor, DNA circuits, DNA storage, drug delivery etc. Tile assembly model of DNA has been studied for various computational primitives such as addition, subtraction, multiplication, and division. Xuncai et. al. gave computational DNA tiles to perform division of a number but the output had integer quotient [1]. In this work, we simply modify their method of division to improve its compatibility with further computation and this modification has found its application in computing rational numbers, both recurring and terminating, with computational tile complexity of $O(1)$ and $O(h)$ respectively. Additionally, we also propose a method to compute square-root of a number with computational tile complexity of $O(n)$ for an n bit number. Finally, after combining tiles of division and square-root, we propose a simple way to compute the ubiquitously used irrational number, π , using its GregoryLeibniz infinite series.

1 INTRODUCTION

In the last two decades, DNA has been used for archival data storage [2], molecular logic circuits [3], building complex 2D structures from single stranded DNA [4] as well as synthetic 3D polyhedra [5]. This high utility of DNA is because of its known nano-scale structure, programmable nature, and flexibility [6]. Using double crossover (DX) and triple crossover (TX) molecules, Winfree showed that DNA self-assembly is Turing Universal [7, 8, 9, 10]. To approximate self-assembly of DNA, he also proposed two models - abstract Tile Assembly Model (aTAM) and kinetic Tile Assembly Model (kTAM). Many well-known problems have been solved using tile assembly model [11, 12, 13, 14, 15]. In particular, tile-assembly models to perform arithmetic computations such as addition, multiplication and factor were shown by Yuri Burn [13, 14, 15]. Addition and multiplications operations had computational tile complexity of $O(1)$ and running complexity of $O(n)$ [14] where as the factoring numbers nondeterministically had $O(1)$ distinct components [15]. Similarly, Xuncai et. al. also came up with computational tiles for subtraction and division of integers, again, using $O(1)$ computational

tiles. The running time complexity is $O(n)$ for subtraction and $O(n^2)$ for division [1]. Additionally, open source softwares such as XTile (it can convert any computational tile formula to a .tile file supported XGrow [16]), ISU TAS [17] and XtileMod (it can generate required computational tiles for performing arithmetic operations as well as the corresponding .tiles file for DNA tiles) have been developed [18, 19].

In the present paper, we propose a method to compute rational number, irrational number(π), and square-root of a number. Additionally, we also propose generalized method to compute division by extending Xuncaï's method. In [1], the output of the division was in the form of quotient and remainder which, however, we have changed to decimal form. This paper could be useful for creating DNA based calculator.

The paper is organized as follows. In Section 2, we briefly introduce aTAM. Section 3 provides basic computational tiles of Xuncaï et. al. division. This tile systems are also used in computing square-root, rational number and π with some minor modifications, in subsequent sections. In Section 4, we provide computational tiles to perform division in rational form. In Section 5, we have given an interesting additional tile system, the insertion tile system, required to compute square-root. Also, we show how to compute square-root. In Section 6, we have given method to compute rational number and π as application of the square-root tile system. In Section 7, we briefly tell about our command line tool which can be used to generate .tile file for computing square-root of a number. Section 8 concludes paper with general remark.

2 BACKGROUND

In [7], Winfree gave abstract Tile Assembly Model (aTAM) and kinetic Tile Assembly Model (kTAM) to understand the relation between self-assembly and computation. The basic building block of aTAM is a unit square tile. The edges of these tiles are nothing but sticky ends of DNA which can combine with other sticky ends of same edge label or edge color. Every tile has 4 edges - North, South, East and West - with a corresponding edge color and glue strength. Also, it is to be noted that, a tile cannot be rotated. In this paper, most of the symbols used have identical meaning to those used in [1]. Mathematically, a tile system is represented as $S_R = (T, S, g, \tau)$, as shown in Fig. 1, where \mathbf{T} is the tile system, \mathbf{S} is the seed configuration of this tile assembly, \mathbf{g} is the glue strength of an edge, and τ (always ≥ 0) is the threshold temperature. The tile assembly starts growing from seed tile and input tiles at the bottom of the tile assembly system. A new tile will attach to the current tile if :

- Edges have same glue color or glue label.
- The combined glue strength of both the tiles is greater than or equal to the threshold temperature (τ).

For a given a seed configuration, we can represent above mentioned conditions in mathematical form. A tile t can attach at point (x, y) if following condition is satisfied [1]:

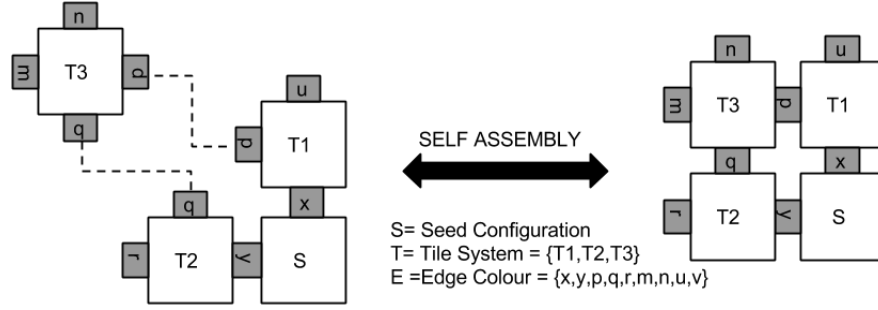


Fig. 1: The image shows basic concept for self-assembly of tiles.

$$\sum_{d \in D} g(bd_D(t), bd_{D^{-1}}(S_R(x, y))) \geq \tau \quad (1)$$

Here, D indicates set of four directions i.e $\{N, E, S, W\}$ and D^{-1} indicates direction opposite to D . $bd_D(t)$ indicates binding domain of tile t in direction D . $g(a, b)$ indicates combined glue strength of glue colors a and b . Given this set of condition, a tile assembly will start growing from initial tile set configuration. Boundary tiles with one edge of glue strength zero are used to avoid any further growth of tiles. We represent a number $a = \sum_{i=0}^{n-1} a^i 2^i$ by $a_0 a_1 a_2 a_3 \dots a_{n-1}$ where a_0 is the most significant bit and a_{n-1} is the least significant bit. For more details, the reader should refer to [1, 7].

3 TILE SYSTEMS

Square-root is function which is provided, almost, by every existing calculator. However, in order to compute square-root using DNA tiles, we propose to use four different kinds of tiles systems, which are similar to those provided by [1], however, with some modifications. The similar tile systems will also be used to compute rational numbers and π .

1. Comparator Tile System
2. Shift Tile System
3. Shift and Subtract Tile System

3.1 COMPARE TILE SYSTEM

Compare tile system [1] helps us to identify relationship between two numbers a and b that is whether a is $\{>, < \text{ or } =\}$ than b . As shown in the Fig. 2, a basic comparison tile consists of 12 different tiles where $\mathbf{R} = \{<, > \text{ or } =\}$ depending on the relationship between $a_0 a_1 a_2 \dots a_{k-1}$ and $b_0 b_1 \dots b_{k-1}$. For example, if $a = 10111_2$ and $b = 11000_2$ then, as shown in Fig. 3, we take input tiles, S_0 , and CL

in seed configuration which grows to unique configuration as shown in Fig. 3. Here, since $a_0 = b_0$ so $\mathbf{R} = \{=\}$ for first bit. For second bit, $a_1 < b_1$ so $\mathbf{R} = \{<\}$. Once we get a $\{>\}$ or $\{<\}$, we continue this sign till the end of system since irrespective of coming bits since the first number is already proved $\{>\}$ or $\{<\}$ than other. This proved that $a < b$ so all the succeeding tiles will have same \mathbf{R} . The proof that compare tile system is a step configuration, total $k + 1$ steps so $O(k)$, and producer of unique tile system for the given seed configuration has already been given in [1].

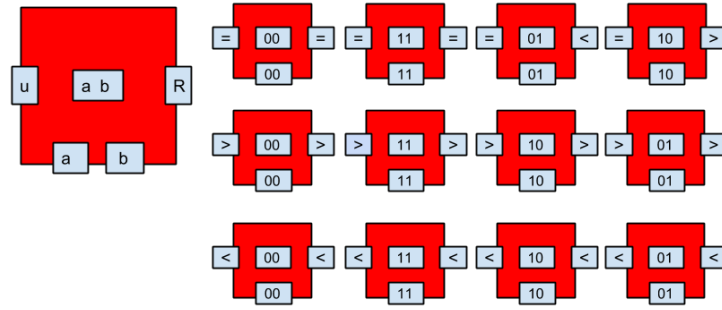


Fig. 2: Comparator System: Left Tile is the general tile representation for all the 12 comparator tiles. Here $a, b, c \in \{0, 1\}$.

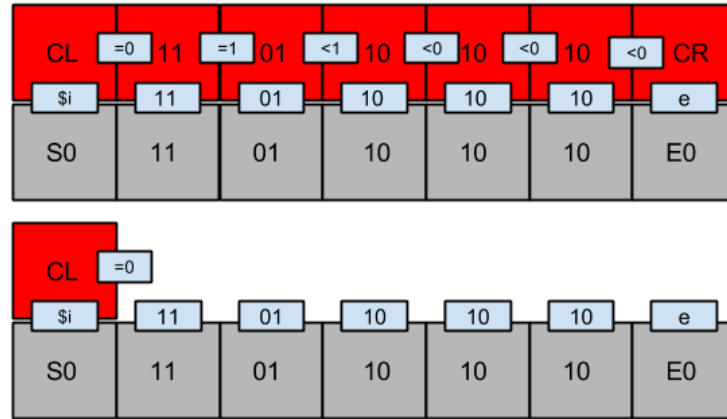


Fig. 3: The image shows how compare tile system grows from seed configuration for $a = 10111_2$ and $b = 11000_2$.

3.2 SHIFT TILE SYSTEM

Shift tile system [1] helps us to right shift the bits of a k bit number by one place while also padding a 0 in most significant bit. As shown in Fig. 4, the system has input bit a , b , and c coming from the south end of tile. Here, a and b represent the input bits and c represents the right bit of previous tile (left tile). For example, as shown in Fig. 5, for input $a = 10111_2$, seed configuration includes $S0$, SL , and input tiles. This seed configuration gives unique final output $a = 01011_2$, as shown in Fig. 5, where a is shifted right by one bit. The proof for the shift tile system is given in [1] as the Duplicator Tile System. This seed configuration also produces unique final output and is a step configuration with time complexity $O(k)$ for a k bit input number.

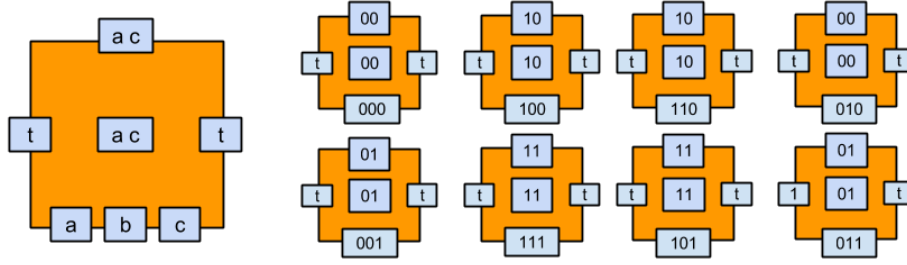


Fig. 4: Shift System: Left Tile is the general tile representation for all the 8 shift tiles.

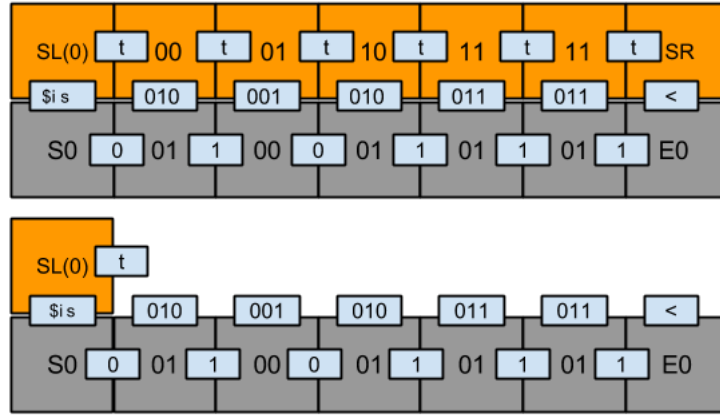


Fig. 5: The image shows how shift tile system grows from seed configuration for $a=10111_2$. The number a , which is to be shifted, is right bit of input tiles.

3.3 SUBTRACT & SHIFT TILE SYSTEM

Subtract & shift tile system [1] is used to subtract right input bits (divisor) from left input bits (divident) and then shift the right bits by one. Fig. 6 shows computational tiles for subtracting and shifting a number. Here, a, b, c, and d are inputs where b is current right bit and c is the right bit of previous tile. This c bit, therefore, helps to right shift the number and xor operation helps in subtraction. For example, as shown in Fig. 7, let $a=10111$ and $b=01100$. Hence, growth of this tile system gives $a - b = 01011$ as left bits and right shift version of b as right bits. Again, the proof and details of this method are provided in [1], so we will not go in to details here. But this tile system is also step configuration with time complexity of $O(k)$ taking total of $k + 1$ steps for a k bit input number.

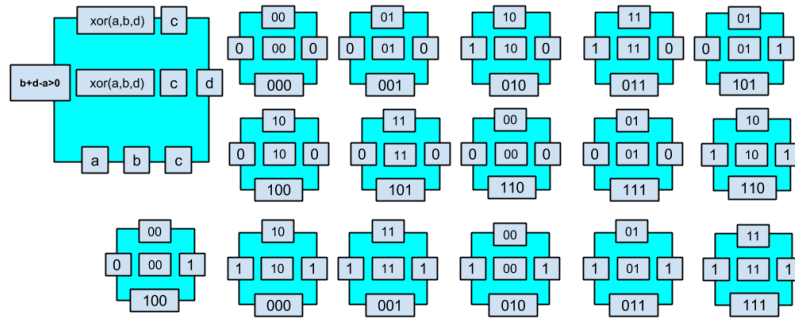


Fig. 6: Shift and Subtract System: Left Tile is the general tile representation for all the 16 shift tiles.

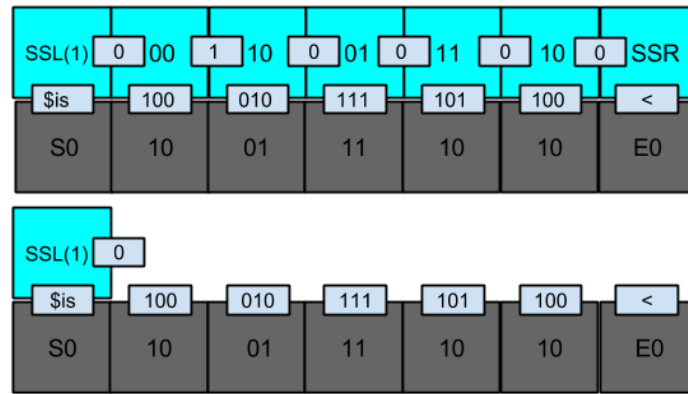


Fig. 7: The growth of subtract and shift tile system from seed configuration for $a=10111$ and $b=01100$.

4 EXTENDING DIVISION

Earlier, Xunca et. al. had proposed a method to compute division using DNA tiles [1]. However, their idea of division considers dividing dividend by divisor to get quotient and remainder. Instead, here we propose a method to get output in different form. We give quotient in decimal form so that one can use this decimal quotient, subsequently, in doing further calculations using tile assembly. Computing division is divided in two cases as shown in the Fig. 8.

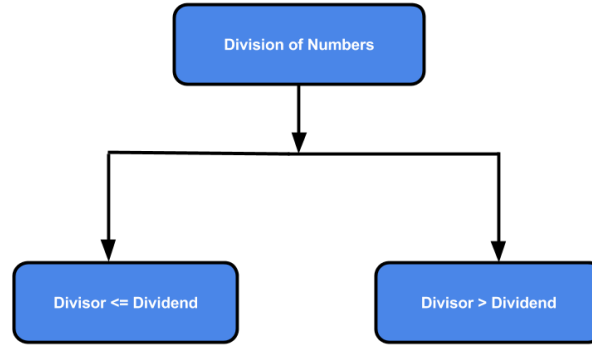


Fig. 8: Flow chart showing both the cases for computing division of a number.

We have to see whether divisor is greater than dividend or vice versa. If the divisor is greater than dividend then we write dividend bits from its right end and prepend zero's, as shown in Fig. 10. In the other case, we write divisor bits from its left end and append zero's for the remaining bits, as shown in Fig. 9. In the paper, we give an example of division where quotient turns out to be infinite.

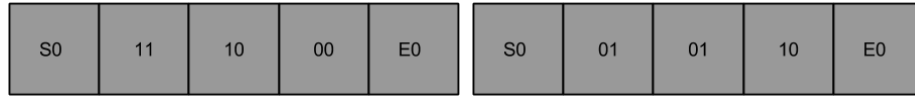


Fig. 9: Divident = 6 and divisor = 1. Fig. 10: Divident = 1 and divisor = 6.

Fig. 11 shows the tile assembly growth for division of 23.5 by 6. The answer to this division is $3.91666666\ldots$ - a recurring rational number. Suppose we want first eight seven bits of the number. In that case, we have to give eight bit input to the tile assembly so that we get 5 bits before decimal place and 3 bits after decimal place. Note that this division is an example of case 1 as depicted in Fig. 9 but the number of bits are, inadvertently, same for both divisor and dividend.



Fig. 11: Division of $a = 23.5$ by $b = 6$ is calculated by this self-assembly. Here a is encoded in binary as 10111.1_2 and b is encoded as 110_2 . We get 11.1110_2 as the output at the left side of the tile assembly model. Yellow tiles simply indicates decimal point. Grey tiles indicate values after decimal point.

Note that the *dot* tiles used in the division process are optional. They are just for better understanding of the reader. Also, the square dots in the Fig. 11 indicate that the assembly can be extended further if we want more answer bits.

The SSL and SL bits represent our answer - quotient - when seen in bottom-up fashion. In this case, we connect $0111110 \dots_2$ from the left boundary tiles and we place a decimal point after three bits which gives us $011.1110_2 = 3.916$. After we get eight bits of quotient, we can terminate growth of this tile assembly with the help of boundary tiles. To sum up, we have not used any additional tiles to compute quotient in decimal form that is these are the exact same tile used in [1]. Thus, the computation uses $O(1)$ computational tiles and this computation takes $O(k^2)$ steps for a k bit number. However, if we wish to terminate the computation then we will require $O(h)$ boundary tiles as shown in Fig. 11 where h is the height of assembly.

5 SQUARE-ROOT

In this section, by combining the tile systems mentioned in Section 4, we illustrate how to find square-root of a number. We will describe a new tile system required to compute square-root of a number.

5.1 INSERT BIT TILE SYSTEM

This tile system is used to insert a bit in a number. Fig. 12 shows the computational tiles required to do this operation. As shown in the Fig. 12, a , b , c , and $\#i$ are the inputs. The values on the east side are output values so this tile system grows from west to east. Here, we subtract the value of i unless it becomes zero and as soon as it does, we stop. We do this since we want to insert bit at i^{th} position in the number. So, as soon as i becomes zero, we place the value of c there and after this tile, we shift all the remaining bits by one to right. For example, as shown in Fig. 13, to insert 1 at 5^{th} position in 100100_2 , we take $\#41$ as the east input of IL which is reduced unless 4 becomes 0 to insert 1 as right bit at that position and shift later bits by one place.

Let us take an example number $n = 100100_2$. Suppose, we want to insert number 1 at fifth position. Therefore, as shown in Fig. 13, $bd_N(S(-k, -1)) = (\#4, 1)$. Now, in the second level, we keep on reducing the $\#i$ value unless we get zero. As soon as we get a zero, we insert the c bit at that position. After this point, we simply right shift the remaining values.

Theorem 1 *Let Σ be an alphabet defined as*

$$\Sigma = \{00, 01, 10, 11, l\} \cup \{\#ic | 1 \leq i \leq n-1 \text{ and } i = 2k+1 ; c \in \{0, 1\}\}$$

and T_1 be the set of tiles defined over Σ . See Fig. 12. let $g = 1$, $\tau = 2$, and S be a seed configuration as defined in Fig. 13. Let q be the number to insert the bit and let m and n be the sizes, in bits, of p and q , respectively. If $m < n$, the number q needs to be padded to be m bits long with extra 0 in the q s high bit. Let $S_R = (T_1, S, g, \tau)$. Then, there exists some $(x_0, y_0) \in \mathbb{Z}_2$ such that: $S(x_0 + 1, y_0 - 1) = E_0$, $S(x_0 - k, y_0 - 1) = S_0$, $S(x_0 - k, y_0) = IL$, $S(x_0 + 1, y_0) = IR$; for all $0 \leq i \leq k-1$, $bd_N(S(x_0 - i, y_0 - 1)) = p_i q_i$. For all other positions (x, y) , $S(x,$

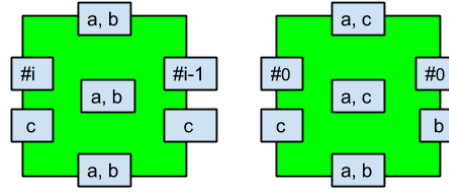


Fig. 12: Insert Tile System: General Tiles for inserting a bit at i^{th} position from left side. Here $a, b, c \in \{0, 1\}$ and $1 \leq i \leq n - 1$.

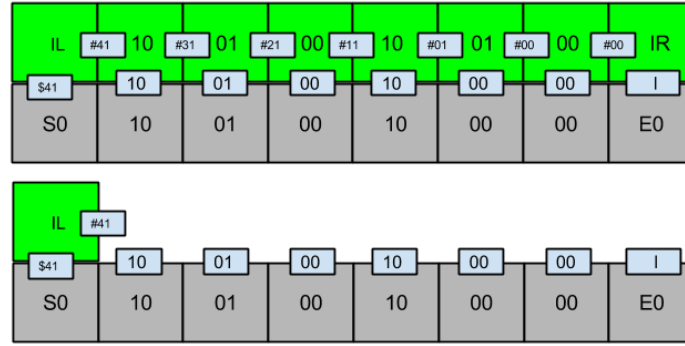


Fig. 13: The assembly inserts 1 at the 5^{th} position from left in $a = 100100_2$.

$y) = \text{empty}$. Then, S produces a unique final configuration F on S to insert a bit at i^{th} position in q . The assembly configuration time is $O(m)$.

Proof. Consider the tile system S_R . Let input tiles be $T_1 = \{E_0 = \{e, \text{null}, \text{null}, \text{null}\}, D_{00} = \{00, \text{null}, \text{null}, \text{null}\}, I_L = \{\text{null}, \#ix, \$ix, \text{null}\}, D_{01} = \{01, \text{null}, \text{null}, \text{null}\}, D_{10} = \{10, \text{null}, \text{null}, \text{null}\}, D_{11} = \{11, \text{null}, \text{null}, \text{null}\}, S_0 = \{\$ix, \text{null}, \text{null}, \text{null}\}\}$.

Let the seed configuration $S : Z_2 \rightarrow T_1$ be:

$$\begin{cases} E_0 = S_R(1, -1) \\ S_0 = S_R(-k, -1) \\ IL = S_R(-k, 0) \\ \forall i \in \{0, 1, \dots, k-1\}, S_R(-i, -1) = D_{pq} \\ \text{For all other } (x, y) \in Z_2, S_R(x, y) = \text{empty} \end{cases} \quad (2)$$

For the given seed configuration S as shown in Fig. 13, there is only one position where the incoming tile can attach. Also, given that $\tau = 2$, by induction we can say that $\forall t \in T_1, (bd_S(t), bd_W(t))$ is unique. Therefore, it follows that S_R produces unique final configuration F and it takes $k + 1$ steps to assemble.

As shown in the Fig.12, value of $\#i$ indicates whether the bit needs to be inserted or simply copied to the other side. If $(\#i, c)$ has $i > 0$ then the output

on east will be $(\#(i-1), c)$. If, on the other hand, we have $(\#0, c)$ as the input on west side, then we will replace (a, b) by (a, c) and the output on east side will be $(\#0, b)$.

Let the final configuration be F . Also, S and F agree on the points $(-i, -1)$, $(-k, -1)$ and $(-k, 0)$ for all $0 \leq i \leq k-1$. Therefore, $bd_N(F(-i, -1)) = pq$. From the definition of S_R , $bd_W(F(-k, 0)) = (\#i, c)$ and the $bd_W(F(-(k-1), 0)) = (x, y)$ where y depends on the value of x . If $\exists m \leq k-1$ where $\#i = \#0$, and for all $n > m$, $i > 0$, then for all $m > n$ we have $bd_E(F(-n, 0)) = (\#(i-1), c)$ and for all $0 \leq n \leq m$, we have $bd_E(F(-n, 0)) = (\#0, b)$ and $v(-n, 0) = (x, c)$. After the relationship is determined and right boundary tile IR is attached at $(1, 0)$, the insertion process will terminate. Additionally, we have $\tau = 2$ which means that only tile will attach at a time; the one which has two neighboring tiles. To sum up, this clearly indicates that we will have unique tile configuration which takes $k+1$ steps to finish ■

5.2 SQUARE-ROOT ASSEMBLY

Suppose we wish to find square-root of a number then we write its bits on left side in input tiles. If the length of n is odd then we add an extra zero bit in the leftmost tile and then start filling the bits. The right bits of the input tile system has 0 and 1 in the two leftmost tiles respectively and all other tiles have zero in the right part. Fig. 15 show the input configuration for $n = 42.25$. In this system, there are four operations namely compare (red tiles), shift (orange tiles), subtract and shift (blue tiles), and insert bit (green tiles). Compare and Insert operations have assembly growth from west to east while the others have it from east to west. As shown in Fig. 15, we compare input numbers, $p = 101010.01$ and $q = 010000.00$, from west to east and since $p > q$ subtract and shift tile system attaches. After the growth of this tile system, we insert 1 at 2^{nd} position from left. Now, the process is repeated that is compare, shift or subtract and shift, and insert perpetually unless the northern glue of insertion tiles becomes equal to $\$(n-1)$ and then the termination end tiles self assemble themselves. At any point if compare tile give $<$ as output then instead of subtract and shift boundary tile, shift boundary tile will attach to CR . The boundary tiles for all these tile systems are shown in Fig. 14.

Note that by rational number we mean that the number should be recurring or terminating. However, in the recurring case, like calculator, we take first 8 bits after decimal.

Theorem 2 Let Σ be an alphabet defined as

$$\begin{aligned} \Sigma = & \{0, 1, 00, 01, 10, 11, e, l, t, <, >, =, r, dot\} \cup \\ & \{xy \mid x \in \{<, >, =\}; y \in \{0, 1\}\} \cup \\ & \{000, 001, 010, 011, 100, 101, 110, 111\} \cup \\ & \{\$i \mid -1 \leq i \leq n-1; i = 2k+1\} \cup \\ & \{Xi \mid 1 \leq i \leq n-1; i = 2k+1\} \cup \\ & \{\#ib \mid 1 \leq i \leq n-1; b \in \{0, 1\}\} \cup \\ & \{\$ix \mid -1 \leq i \leq n-1; x \in \{0, 1, s\}\} \end{aligned}$$

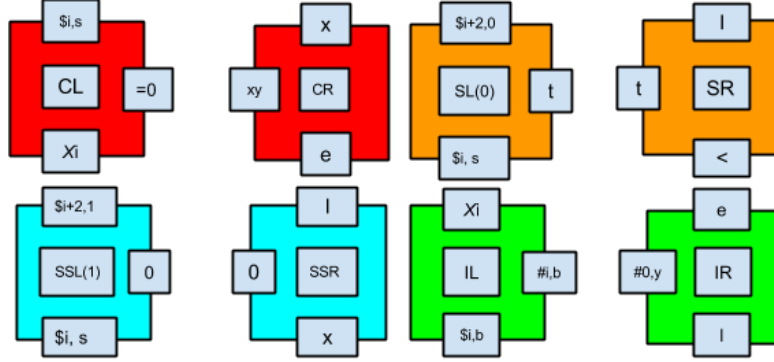


Fig. 14: Boundary tiles for compare, shift, insert, and subtract & shift (clockwise). Here, the variables used in the glue colors of boundary tiles are : $\{\$ix | -1 \leq i \leq n-1; x \in \{0, 1, s\}\}$, $\{Xi | 1 \leq i \leq n-1; i = 2k+1\}$, $\{\#ib | 1 \leq i \leq n-1; b \in \{0, 1\}\}$, $\{xy | x \in \{<, >, =\}; y \in \{0, 1\}\}$, $\{x | x \in \{>, =\}\}$.

and T_2 be the tile set defined over Σ . See Table 6.1, given that $\tau = 2$ and $g=1$, can compute square root of a n bit rational number defined as $S_R = (T_2, S, g, \tau)$ where S is the seed configuration.. Then, there exists some $(x_0, y_0) \in Z_2$ such that : $S_R(x_0 - k, y_0 - 1) = S_0$, $S_R(x_0, y_0 - 1) = E_0$, $S_R(x_0 - k, y_0) = CL$, and for all $i \in \{0, 1, \dots, k-1\}$, $bd_N(x_0 - i, y_0 - 1) = pq$. Then, the configuration S can produce unique final configuration F and will compute square-root of a number.

Proof. Consider the tile system S_R . Let input tiles be $T_2 = E_0 = \{e, null, null, null\}$, $D_{00} = \{00, null, null, null\}$, $D_{01} = \{01, null, null, null\}$, $D_{10} = \{10, null, null, null\}$, $D_{11} = \{11, null, null, null\}$, $S_0 = \{\$ix, null, null, null\}$.

Let the seed configuration $S : Z_2 \rightarrow T_2$ be:

$$\begin{cases} E_0 = S_R(1, -1) \\ S_0 = S_R(-k, -1) \\ CL = S_R(-k, 0) \\ \forall i \in \{0, 1, \dots, k-1\}, S_R(-i, -1) = D_{pq} \\ \text{For all other } (x, y) \in Z_2, S_R(x, y) = empty \end{cases} \quad (3)$$

For the given seed configuration S as shown in Fig. 13, there is only on position where the incoming tile can attach. Also, given that $\tau = 2$, by induction we can say that $\forall t \in T_2$, $(bd_S(t), bd_W(t))$ and $(bd_S(t), bd_E(t))$ is unique. Therefore, it follows that S_R produces unique final configuration F and it takes $k+1$ steps to assemble.

Let the final configuration be F . Also, S and F agree on the points $(-i, -1)$, $(-k, -1)$ and $(-k, 0)$ for all $0 \leq i \leq k-1$. Consider the seed configuration as shown in Fig. 3. According to Lemma 1 of [1], we will get unique configuration for the

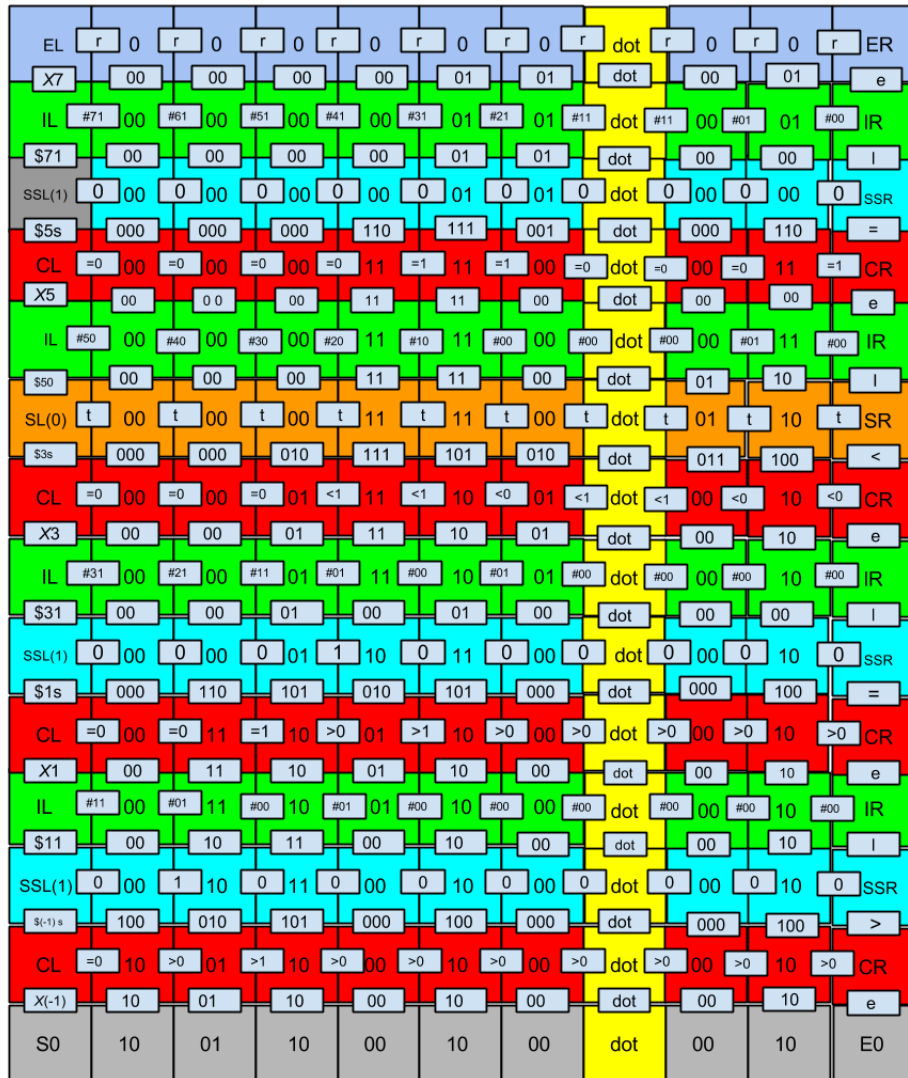


Fig. 15: Square Root of $n = 42.25$ is calculated by the self-assembly of tiles. Here n is encoded in binary as 101010.01_2 . We get 110.1_2 as the output at the left side of the tile assembly model. Color of the tile represent same tile-set as mentioned. Yellow tiles simply indicates decimal point. Grey tiles indicate values after decimal point.

comparison tile system which will terminate with CR attaching at $(1, 0)$ position. Now, if $bd_N(CR)$ is $=$ or $>$, then only SSR will attach to this configuration since

it is the only matching tile. Now, we have new seed configuration which resemble to subtract and shift seed configuration. Again, according to Lemma 3 of [1], this will produce unique final configuration in $k+1$ steps which terminates when SSL tile attaches at $(-k, 0)$. If on the other hand, $bd_N(CR)$ is $<$ than SR tile will attach. In the case also, according to Lemma 2 of [1], we will get a unique tile configuration in $k+1$ steps. This process terminate when SL will attach at $(-k, 0)$ position. After either of this configuration is formed, IL will attach at $(-k, -1)$ position since we have $\tau = 2$ which makes it the only possible tile to be attached to this growing assembly. According to Theorem 1, this insertion seed configuration will produce unique output tile assembly.

Now, the same process repeats unless we get the desired number of bits after decimal place. Therefore, according to induction, the entire configuration will be unique. After all the computation is finished, the corner tiles and boundary tiles will attach at the top to stop the growth of assembly. Hence, square-root computation is terminated ■

Theorem 3 *The time taken for computing square-root is $O((k_1 + k_2)^2)$ where k_1 is number of bits before decimal and k_2 is number of bits after decimal.*

Proof. In our previous proofs, we have already shown that all the tile assemblies are step configuration which means that only one tile is attached to the seed configuration at a time. Also, since $\tau = 2$ no tile will get attached to the configuration unless it has two surrounding tiles. Also, our tile assembly takes k_c levels to compute and in each level we have $n = k_1 + k_2$ computations. This makes total computation time to be equal to $k_c * (k_1 + k_2)$ where $k_c = 3n/2$. Therefore, since $n = k_1 + k_2$, we have running time complexity as $O((k_1 + k_2)^2)$ ■

Basic Tiles	Tile Types
Left Frame	$2n$
Right Frame	11
Corner	4
Top Frame	4
Input	4
Computational	$12 + 8 + 16 + 4n$

Table 1: Tile Set to compute square-root of n bit number. The number n is of even length and if not then made even.

6 APPLICATIONS

The method of square-root can be used to compute rational numbers. Also, a slight modification can help us to compute an irrational number π .

6.1 COMPUTING RATIONAL NUMBERS

Suppose we are interested in finding the square-root of $1/3$ then, first, we need it in decimal form. Therefore, we propose a method to compute rational number

p/q given two input integers p and q . It is an infinitely growing tile assembly which is bounded from west and south, the other ends being open. The method to compute rational number requires some new additional computational tiles but, fortunately, this method will not require the boundary tiles. Fig. 16, 17, and 18 shows additional tiles required to make an infinite assembly growth. The growth for computing $1/3$ is shown in Fig. 19.

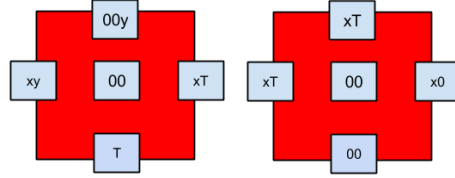


Fig. 16: Additional 9 computational tiles required for comparing bits. Here $x \in \{<, >, =\}$ and $y \in \{0, 1\}$.

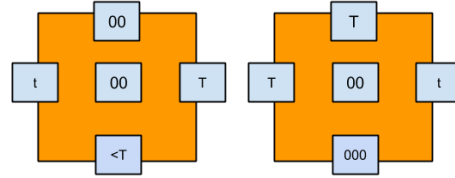


Fig. 17: Additional 2 computational tiles required for shifting divisor bits by one position.

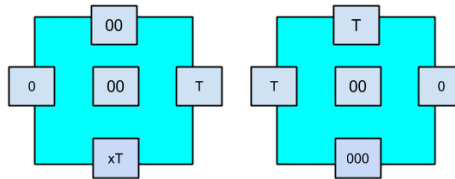


Fig. 18: Additional 2 computational tiles required for subtracting and shifting bits. Here, $x \in \{>, =\}$.

Theorem 4 Let Σ be an alphabet defined as

$$\Sigma = \{0, 1, 00, 01, 10, 11, T, t, <, >, =\} \cup \{xy \mid x \in \{<, >, =\}; y \in \{T, 00, 01, 10, 11\}\} \cup \{000, 001, 010, 011, 100, 101, 110, 111\}$$

and T_3 be the tile set defined over Σ . See Table 6.1, given that $\tau = 2$ and $g=1$, can compute a rational number from a given numerator and denominator. Therefore, the tile assembly is defined as $S_R = (T_3, S, g, \tau)$ where S is the seed configuration.. Then, there exists some $(x_0, y_0) \in \mathbb{Z}_2$ such that : $S_R(x_0 - k, y_0 - 1) = S_0$, $S_R(x_0, y_0 - 1) = 00$, $S_R(x_0 - k, y_0) = CL$, and for all $i \in \{0, 1, \dots, k - 1\}$, $bd_N(x_0 - i, y_0 - 1) = pq$. Then, the configuration S can produce a unique and unbounded final configuration F and will compute rational number.

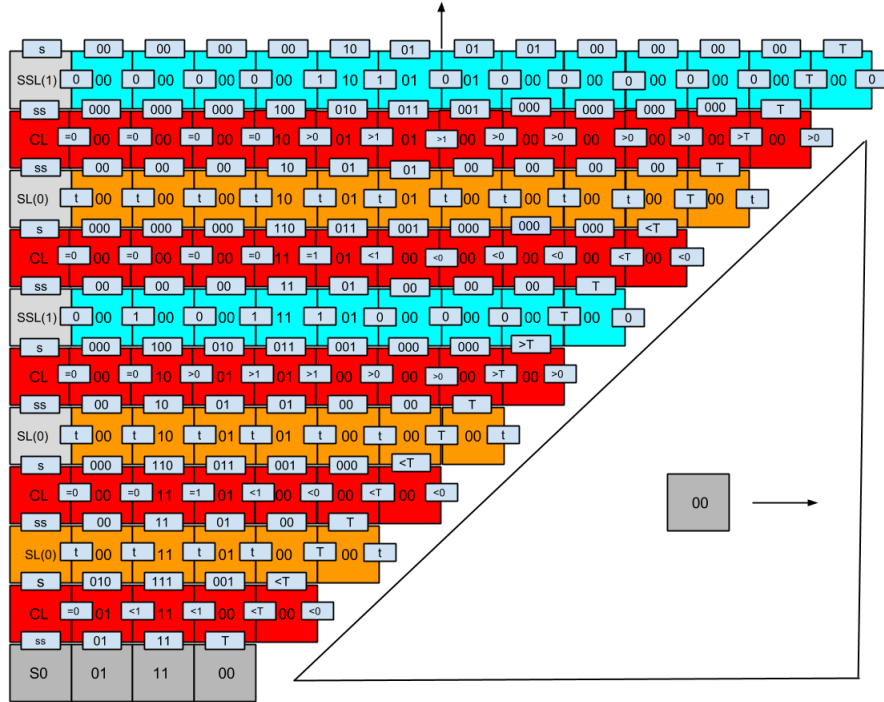


Fig. 19: Here we divide $p = 1$ by $q = 3$ to calculate the rational number $0.3333 \dots$. Here p is encoded in binary as 10_2 and q is encoded as 11_2 . We get 0.0101_2 as the output at the left side of the tile assembly model. Arrows indicate the tile assembly is unbounded while triangle indicates that the entire space will be filled with $(0, 0)$ tiles with relevant glues.

Proof. Consider the tile system S_R . Let input tiles be $T =$
 $\{D_{00} = \{00, null, null, null\}, S_L = \{null, =, ss, null\}$
 $D_{01} = \{01, null, null, null\}, D_{10} = \{10, null, null, null\},$
 $D_{11} = \{11, null, null, null\}, S_0 = \{ix, null, null, null\},$
 $00 = \{T, null, null, null\}\}.$

Let the seed configuration $S : Z_2 \rightarrow T$ be:

$$\begin{cases} 00 = S_R(1, -1) \\ S_0 = S_R(-k, -1) \\ CL = S_R(-k, 0) \\ \forall i \in \{0, 1, \dots, k-1\}, S_R(-i, -1) = D_{pq} \\ \text{For all other } (x, y) \in Z_2, S_R(x, y) = empty \end{cases} \quad (4)$$

Let the final configuration be F . Also, S and F agree on the points $(-i, -1)$, $(-k, -1)$ and $(-k, 0)$ for all $0 \leq i \leq k-1$. The seed configuration S for the tile growing tile assembly shown in Fig. 13 resembles the original compare, shift, and subtract/ shift configurations. There is only one position where the incoming tile can attach but, in this case, after the terminating tile has attached we can have growth on both of its sides. Left side of the terminating tile is exactly similar to division seed configuration hence, according to Lemma 4 of [1], it will have unique growth. As far as the growth of right side of terminating tile is concerned, there are only zero tiles with all the glue colors having values like $\{000, 00, 0, < 0, > 0, = 0\}$ depending on their level.

The same process continues perpetually. Therefore, according to induction, the entire configuration will be unique. Therefore, we get an infinite recurring number. If the rational number is terminating then we will get trailing zeroes.

Type of tiles	Number of Tiles
Compare	$12 + 9 = 21$
Shift	$8 + 2 = 10$
Subtract & Shift	$16 + 2 = 18$
Left Boundary	3

Table 2: Tile Set to compute rational number p/q .

6.2 COMPUTING IRRATIONAL NUMBER - π

In the final section of the paper, we will demonstrate a way to compute π using an infinite series. The computation method uses same tile systems - Subtraction, Shift, Duplicator, Insertion - to compute the value of an infinite series. In order to compute the value of π , we have used *GregoryLeibniz* series which is defined as:

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4} = \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots\right)$$

To compute this series, we will require values of all the fractions so that we sum them up. To find the value of fractions, therefore, we can use Theorem 4 which can compute the value of fraction. Once we have the number in decimal form, we simply need to do addition or subtraction to already existing series sum.

We need the value of output of division in simpler form so that we can use it for further computation. To do this, we have used copy and shift tile systems. Refer Fig. 20. The value of quotient on the left boundary of division tile set is extracted by shifting them one bit per level - yellow and orange tiles - as shown in Fig. 20. As a result, when we finish the division process, we have the output value along with existing sum to which we either want to add or subtract this fraction. Now, depending on whether the term is odd or even we need to add or subtract it to the existing sum which is shown by green tiles in Fig. 20. This arithmetic occurs meanwhile we have input tiles - grey colored - for next division being attached on the other side. Once this addition or subtraction is finished, we replace the current sum with this new sum in the grey tile as shown in Fig. 20. This process keeps run in perpetuity. Depending on the number of precision bits we can control the growth of our assembly.

Theorem 5 *Let Σ be an alphabet defined as*

$$\begin{aligned} \Sigma = & \{0, 1, 00, 01, 10, 11, T, t, <, >, =, +, -\} \cup \\ & \{xy \mid x \in \{<, >, =\}; y \in \{T, 00, 01, 10, 11\}\} \cup \\ & \{000, 001, 010, 011, 100, 101, 110, 111\} \cup \\ & \{\$i \mid -1 \leq i \leq n-1; i = 2k+1\} \cup \\ & \{\#ix \mid -1 \leq i \leq n-1; x \in \{0, 1, s, ss\}\} \end{aligned}$$

and T_4 be the tile set defined over Σ . The tile set includes compare, insert, shift, and subtract \mathcal{E} shift tile systems. Given that $\tau = 2$ and $g=1$, can compute an irrational number π . Therefore, the tile assembly is defined as defined as $S_R = (T_4, S, g, \tau)$ where S is the seed configuration.. Then, there exists some $(x_0, y_0) \in \mathbb{Z}_2$ such that : $S_R(x_0, y_0) = S_0$, $S_R(x_0, y_0 + 1) = CL$, and for all $i \in \{0, 1, \dots, k-1\}$, $bd_N(x_0 - i, y_0 - 1) = pq$. Then, the configuration S can produce a unique and unbounded final configuration F and will compute irrational number - π .

Proof. Since we have used the same tile systems - compare, shift, subtract and shift, and duplicator - its proof will be similar to previous theorems. Therefore, we will not go into details of the proof ■

There are a few observations about *GregoryLeibniz* series [20].

- The series converges at a very slow rate [20]; Leibniz's formula converges extremely slowly: it exhibits sublinear convergence. Calculating π to 10 correct decimal places using direct summation of the series requires about five billion terms.
- The series is easy to compute. The numerator of each term is 4 while the denominator increases by 2 at every-step.
- The series does not require multiplication which means we do not require any new tile system.

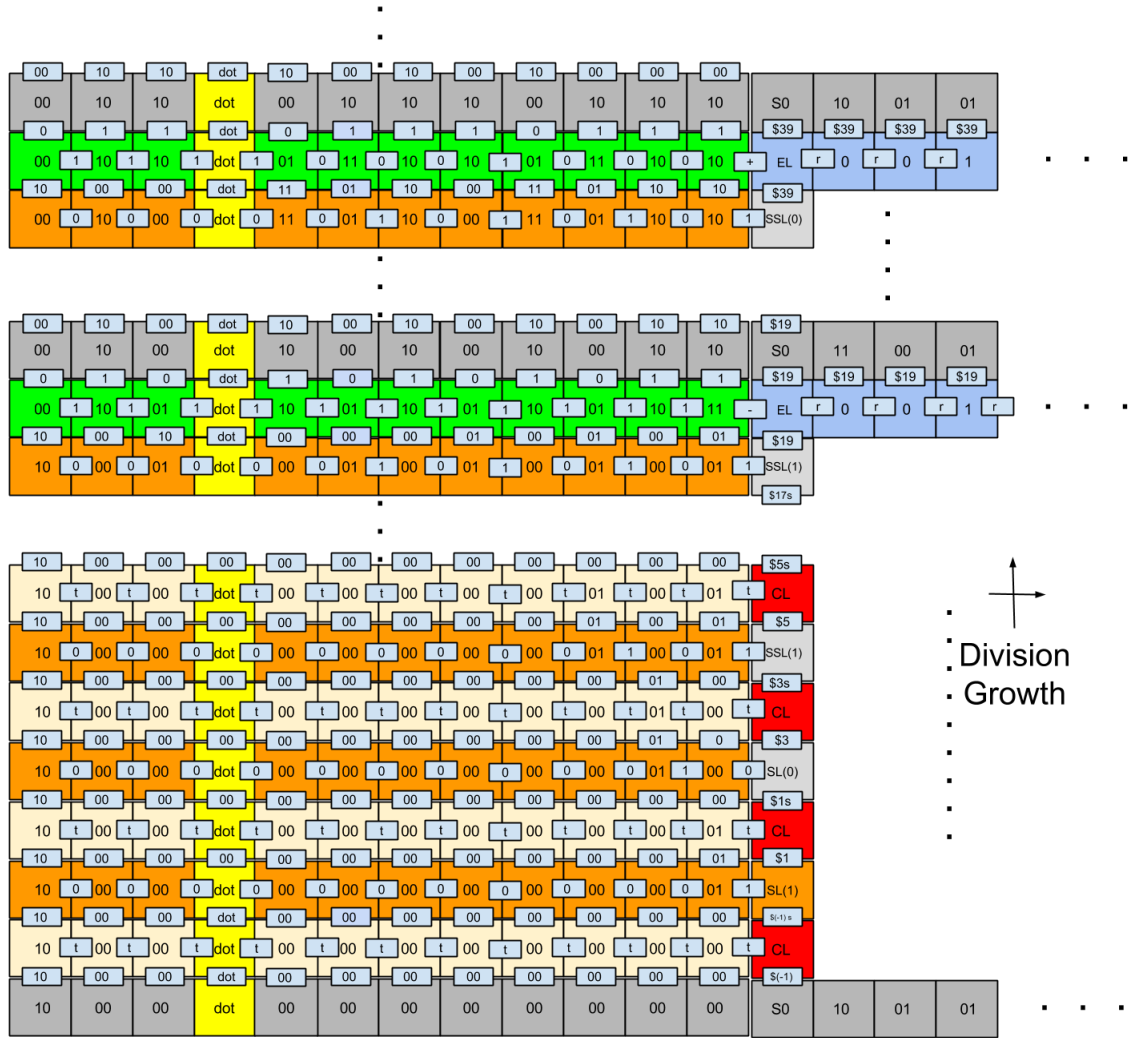


Fig. 20: Computing value of an irrational number $\pi - 3.141592 \dots$. Here, grey tiles are input tiles, green are addition/subtraction tiles, yellow are copy/duplicate tiles, and orange tiles are shift tiles. Blue tiles indicate the remainder of the corresponding division process.

Note that the method to compute π may not be most optimized version. As a result, it is an open problem to compute value of π in an optimized fashion.

7 SUPPLEMENTARY FILE

We have also written a program to generate .tile file of square-root which can be downloaded from our website <http://www.guptalab.org>.

8 CONCLUSION

Our method to compute square-root, and rational numbers is simply a modification of the Xuncai et. al. method to find subtraction and division, using DNA tiles. Our system uses $O(n)$ computational tiles for square-root of an n bit binary number and $O(1)$ tiles for computing rational number. However, it would be an interesting to compute π using finite number of tiles.

References

- [1] X. Zhang, Y. Wang, Z. Chen, J. Xu, and G. Cui, "Arithmetic computation using self-assembly of DNA tiles: subtraction and division," *Progress in Natural Science*, vol. 19, no. 3, pp. 377 – 388, 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1002007108003973>
- [2] S. Shah, D. Limbachiya, and M. K. Gupta, "DNACloud: A Potential Tool for storing Big Data on DNA," *In Proc. of Foundations of Nanoscience Self-Assembled Architectures and Devices(FNANO)*, pp. 204–206, 2014. [Online]. Available: <http://arxiv.org/abs/1310.6992>
- [3] L. Qian and E. Winfree, "Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades," *Science*, vol. 332, no. 6034, pp. 1196–1201, 2011. [Online]. Available: <http://www.sciencemag.org/content/332/6034/1196.abstract>
- [4] B. Wei, M. Dai, and P. Yin, "Complex shapes self-assembled from single-stranded DNA tiles," *Nature*, vol. 485, no. 7400, pp. 623–626, May 2012. [Online]. Available: <http://dx.doi.org/10.1038/nature11075>
- [5] R. Iinuma, Y. Ke, R. Jungmann, T. Schlichthaerle, J. B. Woehrstein, and P. Yin, "Polyhedra Self-Assembled from DNA Tripods and Characterized with 3D DNA-PAINT," *science*, vol. 344, no. 6179, pp. 65–69, 2014.
- [6] C. Lin, Y. Liu, S. Rinker, and H. Yan, "DNA Tile Based Self-Assembly: Building Complex Nanoarchitectures," *ChemPhysChem*, vol. 7, no. 8, pp. 1641–1647, 2006. [Online]. Available: <http://dx.doi.org/10.1002/cphc.200600260>
- [7] E. Winfree, "Algorithmic self-assembly of DNA," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, 1998.
- [8] H. Wang, "Dominoes and the AEA case of the decision problem," in *Computation, Logic, Philosophy*. Springer, 1990, pp. 218–245.
- [9] J. H. Reif, T. H. LaBean, and N. C. Seeman, "Programmable assembly at the molecular scale: self-assembly of dna lattices," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1. IEEE, 2001, pp. 966–971.

- [10] A. Carbone and N. Seeman, "Molecular tiling and dna self-assembly," in *Aspects of Molecular Computing*, ser. Lecture Notes in Computer Science, N. Jonoska, G. Pun, and G. Rozenberg, Eds. Springer Berlin Heidelberg, 2004, vol. 2950, pp. 61–83. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24635-0_5
- [11] L. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 5187, pp. 1021–1024, 1994. [Online]. Available: <http://www.sciencemag.org/content/266/5187/1021.abstract>
- [12] Y. Wang, W. Lu, X. Zhang, and G. Cui, "DNA tile assembly model for 0-1 knapsack problem," in *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, Sept 2010, pp. 180–184.
- [13] Y. Brun, "Solving NP-complete problems in the tile assembly model," *Theoretical Computer Science*, vol. 395, no. 1, pp. 31 – 46, 2008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0304397507006433>
- [14] B. Yuriy, "Arithmetic computation in the tile assembly model: Addition and multiplication," *Theoretical Computer Science*, vol. 378, no. 1, pp. 17–31, 2007.
- [15] Y. Brun, "Nondeterministic polynomial time factoring in the tile assembly model," *Theoretical Computer Science*, vol. 395, no. 1, pp. 3 – 23, 2008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0304397507006287>
- [16] E. Winfree. (2003) The Xgrow Simulator. [Online]. Available: <http://www.dna.caltech.edu/Xgrow/>
- [17] M. J. Patitz, "Simulation of self-assembly in the abstract tile assembly model with isotactic," in *Proc. of Foundations of Nanoscience Self-Assembled Architectures and Devices(FNANO)*, 2009.
- [18] A. Chaurasia, S. Dwivedi, P. Jain, and M. K. Gupta, "XTile: An Error-Correction Package for DNA Self-Assembly," in *Proc. of Foundations of Nanoscience Self-Assembled Architectures and Devices(FNANO)*, vol. abs/0908.2744, pp. 225–229, 2009. [Online]. Available: <http://arxiv.org/abs/0908.2744>
- [19] A. Chhajaj, M. K. Gupta, S. Vasani, and J. Dholakiya, "Modular Arithmetic Expressions and Primality Testing via DNA Self-Assembly," *CoRR*, vol. abs/1207.1161, 2012.
- [20] D. B. Jonathan Borwein and R. Girgensohn, *Experimentation in Mathematics - Computational Paths to Discovery*. Peters, A K, 2003, pp. 28–30.